カーネルにおける Memory Protection Key を用いた権限情報保護機構の提案

葛野 弘樹[1,a)]　山内 利宏[2]

概要：オペレーティングシステムカーネルへの攻撃として，カーネル脆弱性を利用して権限情報を管理者権限に改ざんする特権奪取攻撃が知られている．カーネルにおける攻撃対策として，KCoFI では，カーネルコードの呼出し順序の検査を行い，KASLR はカーネルコード，およびカーネルデータの仮想アドレスのランダム化を行うことで攻撃困難化を実現する．また，AKO では，特権奪取を行う際の権限情報の変化に着目し，意図しないタイミングでの権限情報の変化検知を行い，書き戻す事を可能とした．しかし，既存手法では，攻撃を行うユーザプロセスによる権限情報の書込み制限は行わないことから，依然としてカーネル脆弱性の利用に成功した場合，権限情報の改ざんは可能である．本稿では，既存手法の課題を解決するため，特定のカーネルデータへの書込みを制限し，権限情報の保護に利用可能なセキュリティ機構を提案する．提案するセキュリティ機構では，Memory Protection Key を利用し，カーネルデータの仮想アドレスに対する書込み制限の制御を行う．攻撃を行うユーザプロセスによる，特権奪取時の権限情報の改ざんを防ぐため，実行中のカーネルにおいて，ユーザプロセスに対して，動的に権限情報の書込み制限を実現する．提案手法を Linux にて実現し，評価のため特権奪取に利用可能なカーネル脆弱性を導入した環境において，攻撃を行うユーザプロセスによる権限情報の書込み防止可能性について評価した．

Design and Implementation of Privilege Information Protection Using Memory Protection Key

Hiroki Kuzuno[1,a)]　Toshihiro Yamauchi[2]

Abstract: An operating system provides the essential countermeasures to reduce the effects of kernel vulnerability attacks. Kernel protection methods (e.g., kernel address space layout randomization, control flow integrity, and additional kernel observer) mitigate the malicious activity at the kernel layer. To gain complete control of a host, adversaries focus on the modification of credential information, such as the privilege escalation attack can still occur via the latest kernel vulnerability. This is because the vulnerable kernel code and the attack target kernel data (e.g., credential information) are writable while running the kernel. In this paper, a novel kernel data protection mechanism using Memory Protection Key for supervisor. It protects specific kernel data from adversaries' user processes to manage the read and write permissions for each user process at the kernel layer. The proposed mechanism is realized to the latest Linux that can protect the kernel data (e.g., credential information) against the privilege escalation attack with an acceptable overhead performance at the evaluation.

1. はじめに

オペレーティングシステム（OS）カーネルへの攻撃として，権限情報を管理者に改ざんする特権奪取攻撃が課題とされている．特権奪取攻撃は，攻撃を行うユーザプロセスが脆弱性を含むカーネルコード（以降，カーネル脆弱性）

1 セコム株式会社 IS 研究所
Intelligent Systems Laboratory, SECOM Co., Ltd., Japan
2 岡山大学 学術研究院自然科学学域
Graduate School of Natural Science and Technology, Okayama University, Japan
a) kuzuno@s.okayama-u.ac.jp

© 2021 Information Processing Society of Japan

を利用し，カーネル仮想記憶空間上に配置され，権限情報を格納したカーネルデータに対し，一般ユーザから管理者ユーザに改ざんする攻撃である．攻撃を行うユーザプロセスは計算機の管理者ユーザとして，マルウェアの実行や隠蔽を行い，他の計算機への攻撃を試みる．

カーネル脆弱性を利用した攻撃の防止手法として，KCoFI では，コード呼出し順を検査する control flow integrity（CFI）[1] をカーネルに適用可能とし，不正なコード呼出しを制限した [2]．Kernel address space layout randomization （KASLR）では，カーネルの仮想記憶空間上のカーネルコードやカーネルデータの仮想アドレスをランダム化し，攻撃時の仮想アドレスの特定を困難化した [3]．また，特権奪取攻撃対策として，Additional kernel observer（AKO）は，カーネルにおける権限情報変化に対し，意図しない書替えを検知した場合，元の権限情報に修正し，特権奪取攻撃の防止を実現している [4]．

既存の手法では，ユーザプロセスからの特権奪取攻撃の成立困難化として，脆弱なカーネルコード特定，呼出し，ならびに被害の抑制を行う．しかし，権限情報に対する攻撃自体は可能であり，次の課題があると考えている．

課題： 権限情報の改ざん

特権奪取攻撃では，権限情報の改ざんとして，一般ユーザから管理者ユーザへのユーザ ID の書替えが行われる．従来手法では，権限情報は動作中のカーネルに対して読書き制限の制御は行われない．実行されたカーネルコードには権限情報の書替え制限は行われないことから，攻撃を行うユーザプロセスから脆弱なカーネルコードを実行された場合，権限情報の改ざんを防ぐことは難しい．

本稿では，特権奪取攻撃における，権限情報の改ざんを緩和するため，特定のカーネルデータを保護対象（以降，保護対象カーネルデータ）として，書込み制限を動的に制御するセキュリティ機構を提案する．**図 1** に提案手法の概要を示す．提案するセキュリティ機構においては，ユーザプロセスに対し，予め許可されたシステムコール発行時のみに保護対象カーネルデータへの書込みを許可する．権限情報を保護対象とした場合，権限変更を伴わないシステムコールを介し，攻撃を行うユーザプロセスから権限情報の改ざんの困難化を実現する．

提案するセキュリティ機構においては，保護対象カーネルデータへの書込み制限を Intel Memory Protection Key（MPK）Protection Keys for Supervisor （PKS）を用いて制御する．PKS は，Intel により開発中の MPK の機能であり，現在の MPK はユーザモードに対するメモリ読書き可否を制御できる．PKS はスーパーバイザモード（本稿ではカーネルモードとする）に対し，メモリへの読書き可否を制御する機能であり，Persistemt memory での利用が検討されている [5], [6]．MPK はページテーブルエントリ

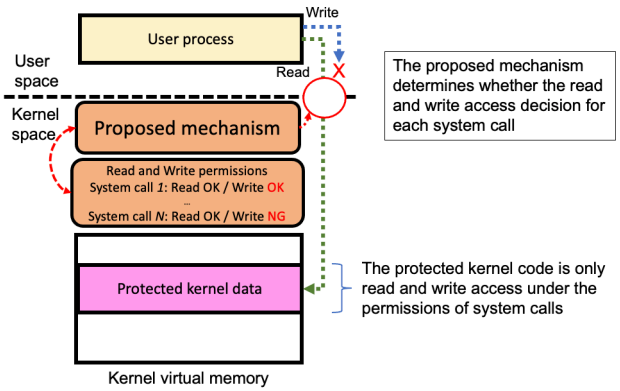

図 1 提案手法の概要

（PTE）毎に Protection key を指定し，読書き制限の設定を可能とする CPU 機能である（詳細は 2.2 節を参照）．PKS をサポートする Intel プロセッサは 2021 年 8 月時点では提供されておらず，次世代 CPU へ実装予定であるが [7]，QEMU 環境では PKS を利用可能である [8]．PKS をカーネルのセキュリティ機構として利用する提案はまだなく，本研究では今後登場する PKS をカーネルの重要なデータの保護に逸早く適用するものである．

提案するセキュリティ機構により，保護対象カーネルデータに対して，PKS を利用した読書き制限の制御が可能になる．権限情報に対して PKS の Protection key （以降，権限情報 Protection key）を指定し，保護対象とすることで，脆弱なカーネルコードが呼ばれた場合においても，権限情報を保護でき，特権奪取攻撃の緩和に繋がる．提案するセキュリティ機構では，権限情報 Protection key による書込み制限をシステムコール発行時に制御し，予め許可されたシステムコール発行時のみ，権限情報の書替えを許可する．これにより，攻撃を行うユーザプロセスにおいて，脆弱なカーネルコードを実行された場合，特権奪取攻撃による権限情報の改ざんが試みられるが，権限情報 Protection key に基づく書込み制限をカーネルが無効にしない限り，権限情報への書込みは成功しない．

本稿での研究貢献は以下の通りである：

（1） カーネル脆弱性を利用した特権奪取攻撃の緩和のため，PKS を利用し，システムコール発行時における権限情報への書込み制限を管理するセキュリティ機構を提案した．動作中のカーネルを対象として，提案手法を実現した Linux にて，セキュリティ機能を検証した．

（2） 提案するセキュリティ機構の評価として，特権奪取攻撃に利用可能なカーネル脆弱性をカーネルに導入し，攻撃を行うユーザプロセスによる権限情報の改ざんを防止可能であることを確認した．また，通常アプリケーションおよびカーネルへの影響可能性を検討した．

2. 背景知識

2.1 カーネル脆弱性を利用した特権奪取攻撃

カーネル脆弱性は，カーネルへの攻撃に利用可能とさ

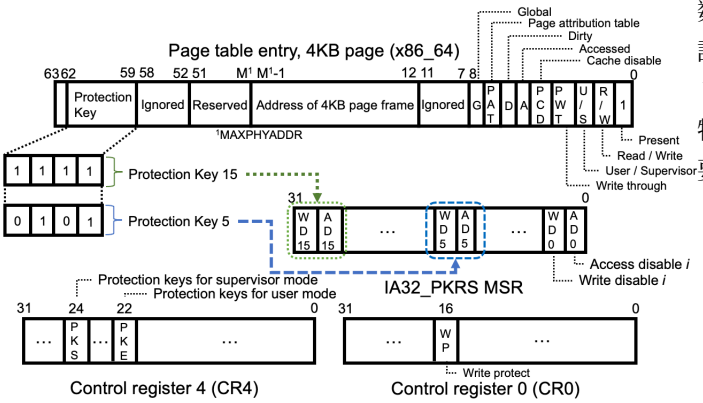

図 2 Intel Memory Protection Key の概要 [9]

れる実装不備とされる [10]．特権奪取攻撃では，カーネル脆弱性を利用して任意のコードを挿入し，カーネルにおける権限操作を行うカーネル関数 `commit_creds` および `prepare_kernel_cred` の呼出しによる権限情報の変更 [11], [12], [13]，または，カーネル仮想記憶空間上の権限情報を格納するカーネルデータの変数 `cred` を改ざんし，ユーザプロセスのユーザ ID を管理者ユーザへ変更する [14]．

2.2 Memory Protection Key

MPK は仮想アドレスを構成する PTE に対して読書き権限を操作可能な Intel CPU の機能である．**図 2** に示す通り，PKS はカーネルモード用の MPK であり，次の機能が提供される [9]．

- Protection keys for supervisor（PKS）：カーネルモードで利用する Protection key である．Control register 0（CR0）Write Protect フラグかつ CR4 PKS フラグにて有効 / 無効を制御する．PKS では，PTE に対し，4 bit 割当てられ，16 個の Protection key を指定可能である．Protection key 毎の読書き制限（Write disable（WD），Read disable（RD））2 bit 操作のため，32bit の読書き制限操作レジスタ IA32_PKRS MSR レジスタ（以降，PKRS）を CPU コア毎に備える．

PKS の i 番目の Protection key i（$0 \leq i \leq 15$）に対して，読書き制限操作レジスタ PKRS の i 番目ビット RDi が 1 の場合，読込み不可．i+1 番目ビット WDi+1 が 1 の場合，書込み不可として CPU に処理される．

従来，PTE 単位での読書き制御は読込みのみ，あるいは読書きの 2 種類を指定できたが，MPK を用いることで読込み制限，書込み制限を個別に制御可能である．また，Protection key を複数の PTE に指定することで，読書き制限操作レジスタのみでの一括制御を可能としている．

3. 脅威モデル

提案するセキュリティ機構の想定する脅威モデルとして，攻撃者の目標は特権奪取とする．攻撃シナリオとして，攻撃者はカーネル脆弱性を利用するユーザプロセスを実行し，脆弱なカーネルコードから権限変更を行うカーネル関数の呼出し，または権限情報の改ざんを行い，特権奪取を試みる．脅威モデルの利用環境において，提案するセキュリティ機構を用いることで攻撃シナリオに沿った攻撃者の特権奪取攻撃の防止を行う．脅威モデルにおける攻撃者の要件，および利用環境を以下にまとめる．

- 攻撃者： 一般ユーザにて，特権奪取可能なカーネル脆弱性を利用する PoC コードをユーザプロセスとして実行．
- カーネル：特権奪取攻撃に利用可能なカーネル脆弱性を含む，既存のセキュリティ機構（例，KASLR，CFI，および AKO）の適用は行わない．
- カーネル脆弱性：攻撃者を行うユーザプロセスより特権奪取に利用可能なカーネル脆弱性．脆弱なカーネルコードから任意の仮想アドレスを指定し，権限変更を行うカーネル関数を呼出し可能．また，権限情報情報を含むカーネルコードへの書込みが可能．
- 攻撃対象： 特権奪取攻撃を行うユーザプロセスの攻撃対象は権限情報の仮想アドレス（例．ユーザプロセスの権限情報）．

4. 提案手法と実現方式

4.1 提案手法の要件

提案するセキュリティ機構では，システムコール毎に動作中のカーネルにおいて保護対象カーネルデータに対して読書き制限を指定するため，設計として次の要件を満たすことを目指した．

要件： 攻撃を行うユーザプロセスによるカーネル脆弱性を利用したカーネルデータの改ざん（例，特権奪取攻撃）を想定し，保護対象カーネルデータの読書き制限を設定可能とする．動作中のカーネルにおける読書き制限の管理として，書込み制限の制御はシステムコール単位で処理する．読書き制限の切替えはシステムコール発行時に行い，書込みを許可されるシステムコールの実行中においてのみ保護対象カーネルデータへの書込み制限を無効可能とする．

4.2 設計

提案するセキュリティ機構の設計概要を**図 3** に示す．要件を満たすため提案するセキュリティ機構では，保護対象カーネルデータ（例，権限情報）を予め指定し，識別子を設けて管理する．また，システムコールと紐付け，保護対象カーネルデータの読書き制限を管理する．

4.2.1 保護対象カーネルデータ

提案するセキュリティ機構における保護対象カーネルデータおよび識別子は次の通りである．

- 保護対象カーネルデータの識別子：読書き制限を設定するために用いるカーネルデータ種別を示す識別子．識別子に対して読書き制限の紐付けと設定を行う．
- 保護対象カーネルデータ：アプリケーションをユーザ

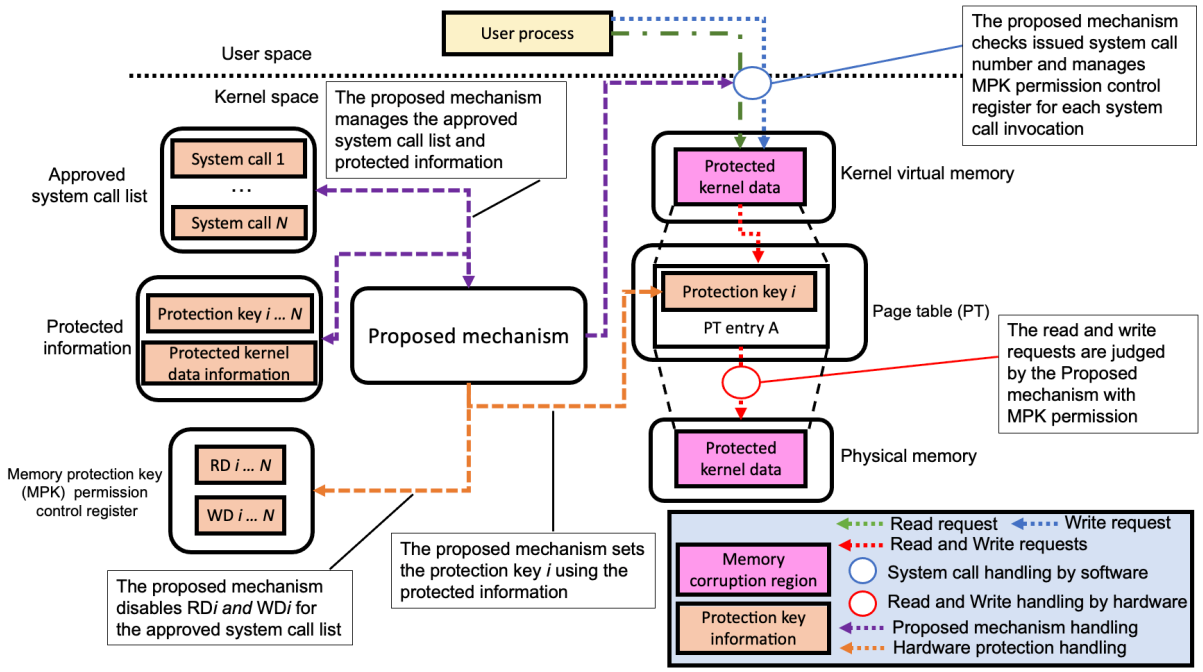

図 3　提案手法の設計概要図

プロセスとして実行する際，カーネルにおいて設定されるカーネルデータ．

提案するセキュリティ機構を備えたカーネルでは，起動時に保護対象カーネルデータと対応する識別子のリストを予め設ける．動作中のカーネルにおいて，保護対象カーネルデータはユーザプロセス毎に生成されることを想定し，ユーザプロセスの起動時にて，識別子との関連づけを行い，同時に読書き制限を設定する．

4.2.2　読書き制限の管理と操作

提案するセキュリティ機構において，保護対象カーネルデータへの読書き制限はシステムコール単位で行う．システムコールを次の 2 種類に分類して管理し，システムコールの発行毎に読書き制限を操作することで取扱う．

- 権限変更許可システムコール：保護対象カーネルデータの書込み権限を有するシステムコール．
- 通常システムコール：保護対象カーネルデータの書込み権限を有さないシステムコール．

提案するセキュリティ機構を備えたカーネルでは，権限変更許可システムコールのリスト（以降，権限変更許可リスト）を管理する．動作中のカーネルにおいて，権限変更許可リストのシステムコール発行時に保護対象カーネルデータの読書き権限を変更する．権限変更許可システムコールの実行時のコンテキストでのみ，保護対象カーネルデータの書込み制限は無効化され，カーネルにおいて権限情報の変更が許可される．通常システムコールの実行時，保護対象カーネルデータへの書込み制限はカーネルにおいて有効化され，書込みは制限される．

4.3　実現方式

保護対象カーネルデータとして，権限情報を想定し，システムコール単位で読書き制限の管理を動作中のカーネルにて行う実現方式を考案した．実現方式にて想定する環境は，x86_64 CPU アーキテクチャの Linux としている．

表 1　保護対象の権限情報カーネルデータ（参考文献 [4] より抜粋）

権限情報	権限情報に関する変数
User ID	uid, euid, fsuid, suid
Group ID	gid, egid, fsgid, sgid

4.3.1　保護対象カーネルデータ

実現方式を備えた Linux カーネルにおいて，保護対象とする権限情報のカーネルデータ（以降，権限情報カーネルデータ）に対し，PKS の機能を次のように設定し，権限情報カーネルデータへの読書き制限制御として利用する．

- 権限情報カーネルデータの識別子：権限情報カーネルデータの読書き制限の管理と操作のため，権限情報 Protection key i とし，PTE の Protection key i（4 bit）の指定値と同値とする．
- 権限情報カーネルデータのページおよび PTE：`task_struct` 構造体の `cred` 構造体に格納される権限情報において，実現方式では，**表 1** に示す権限情報を格納するページ（4KB）を生成し，該当ページの PTE の 59 bit から 62 bit 目に識別子と同値である権限情報 Protection key i を格納する．

ユーザプロセスの生成時，実現方式により，専用ページを生成し，権限情報 Protection key i の指定を PTE に行うことで，ユーザプロセスの動作期間中，提案するセキュリティ機構による読書き制限制御の対象とする．

4.3.2　読書き制限の管理と操作

実現方式において，権限情報 Protection key i に対する読書き制限 RDi，および WDi は PKS の読書き制限操作レジスタ `IA32_PKRS MSR` を用いて管理する．`IA32_PKRS MSR`

表 2　権限変更許可リストのシステムコール（参考文献 [4] より抜粋）

システムコール名	番号	権限情報
execve	59	uid, euid, fsuid, suid, gid, egid, fsgid, sgid
setuid	105	uid, euid, fsuid, suid
setgid	106	gid, egid, fsgid, sgid
setreuid	113	uid, euid, fsuid, suid
setregid	114	gid, egid, fsgid, sgid
setresuid	117	uid, euid, fsuid, suid
setresgid	119	gid, egid, fsgid, sgid
setfsuid	122	fsuid
setfsgid	123	fsgid

は CPU コア毎に備えており，権限変更許可システムコール，および通常システムコール捕捉時の権限情報 Protection key の読書き制限の操作は，次の手順にて行う．

（1）　ユーザプロセスによるシステムコール呼出しを捕捉．

（2）　権限変更許可リストに含むシステムコール番号か判定．

（a）　権限変更許可システムコールの場合：Protection key i に対応する IA32_PKRS MSR の RDi，WDi に 0 を書込む権限情報カーネルデータの読書き許可に設定

（b）　通常システムコールの場合：Protection key i に対応する IA32_PKRS MSR の RDi に 0 を書込み，WDi に 1 を書込む．権限情報カーネルデータの読込みのみ許可，書込みは不許可に設定．

（3）　システムコールの実行を継続する．

権限変更許可リストに含めるシステムコールは，表 1 の権限情報カーネルデータの操作を行うシステムコールとし，**表 2** に示す．

4.3.3　権限情報カーネルデータの保護事例

実現方式を用いた権限情報カーネルデータの保護事例を **図 4** に示す．ユーザプロセスを起動後，権限情報 Protection key 1 を権限情報カーネルデータに設定する．ユーザプロセスの発行したシステムコールが権限変更許可リストに含まれている場合，カーネルにおいてシステムコールの実行中，読書き制限操作レジスタ IA32_PKRS MSR の RD1 および WD1 を 0 とし，権限情報の読書き操作を許可する．一方，システムコールが権限変更許可リストに含まれていない場合，システムコールの実行中，読書き制限操作レジスタ IA32_PKRS MSR の RD1 は 0，WD1 は 1 とし，権限情報の書込みは許可されない．

5.　評価

5.1　評価の目的と評価環境

評価として，提案するセキュリティ機構を適用したカーネルにおいて，特権奪取攻撃から権限情報保護が適切に動作するかの調査を目的とする．セキュリティ機能評価の項目と内容を以下に示す．

（1）　特権奪取攻撃に対するセキュリティ機能の評価

提案するセキュリティ機構を適用したカーネルに対して，特権奪取攻撃に利用可能なカーネル脆弱性を導入し，特権奪取攻撃を行うユーザプロセスを実行する．予め権限情報 Protection key の書込み制限を有効化，特権奪取攻撃実行時に，権限情報の改ざん防止を実現可能か評価した．

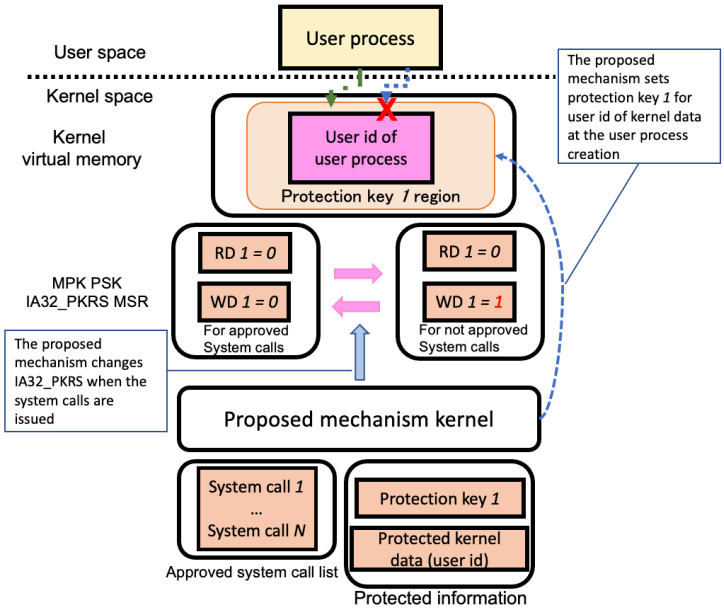

図 4　実現方式を適用した権限情報カーネルデータの保護

2021 年 8 月時点において，PKS を実装した CPU は提供されていない．セキュリティ機能の評価を行う仮想計算機環境は PKS に対応した QEMU 6.0.91 を用いた [8]．QEMU 上のゲスト OS は Debian 10.2 とし，提案手法は Linux kernel 5.3.18 を対象に，15 個のファイルに対して 431 行を追加し実現した．また，特権奪取攻撃を行うための PoC コードとして 41 行を Linux カーネルに追加した．

5.1.1　特権奪取攻撃に利用可能なカーネル脆弱性

提案するセキュリティ機構のセキュリティ機能評価のため，特権奪取攻撃に利用可能な独自システムコールをシステムコール番号 350 としてカーネルに導入した．

- カーネル脆弱性（特権奪取）：独自システムコールを実行中，権限情報を変更するカーネル関数を呼出し，ユーザプロセスの権限情報を管理者ユーザに変更する．

評価において，独自システムコールを呼出す PoC コードをユーザプロセス経由にて実行し，特権奪取攻撃を行い，一般ユーザから管理者ユーザへの変更を試みる．

5.2　特権奪取攻撃に対するセキュリティ機能の評価

評価においては，提案するセキュリティ機構を備えたカーネルにて，導入したカーネル脆弱性を利用する PoC コードを特権奪取攻撃を行うユーザプロセスとして実行する．その際の攻撃開始時から攻撃終了後までのシステムコール，ならびにカーネルにおけるログ出力を行う．

5.2.1　特権奪取攻撃を行うユーザプロセス

図 5 にカーネル脆弱性を利用した特権奪取攻撃を行うユーザプロセスの実行結果を示す．実行時のユーザ ID は

```
1.  $ id
2.  uid=33(www-data) gid=33(www-data) groups=33(www-data)
3.  $ ./kvuln01
4.  [*] malicious user process id
5.  PID: 1661

6.  [*] start setgid (sysnum: 105) system call invocation
7.  [*] start setuid (sysnum: 106) system call invocation
8.  [*] process uid, euid, gid, and egid are changed to 33
9.  uid=33(www-data) euid=33(www-data) gid=33(www-data) egid=33

10. [*] start vulnerable system call (sysnum: 350) invocation
11. [*] process uid, euid, gid, and egid are changed to 0
12. uid=0(root) euid=0(root) gid=0(root) egid=0
```

図 5　導入したカーネル脆弱性を利用した特権奪取攻撃の結果

33，ならびにユーザプロセスのプロセス番号は 1661 である．提案するセキュリティ機構の動作確認のため，ユーザ ID 33 を用いて 6 行目から 7 行目にて，`setuid` および `setgid` システムコールを呼出しを行う．その後，独自システムコールを呼出す．12 行目にて特権奪取攻撃を行うカーネル脆弱性として，ユーザ ID を管理者ユーザ（ユーザ ID 0）に変更するカーネル関数を呼出し，特権奪取に成功した結果が表示されている．

5.2.2　PKS 管理と権限変更許可システムコールの捕捉

図 6 に Linux カーネル起動時の PKS の設定，ならびに権限変更許可リストに含まれるシステムコール捕捉時の動作結果を示す．図 6 の 2 行目にて，PKS （CR4 24 ビット）の有効化を確認できる．5 行目にて，ユーザプロセス生成時に，権限情報カーネルデータの PTE（59 ビットから 62 ビット）への Protection key 1 の設定を確認できる．

図 6 の 6 行目にて，プロセス ID 1661 の権限変更許可システムコール `setgid`（システムコール番号 106）呼出しを捕捉している．12 行目では，PKRS は　0x8 であり，Protection key 1 の書込み制限（WD1）は有効化されている．14 行目にて，PKRS に 0x0 を書込み，Protection key 1 への書込み制限（WD1）を無効化し，権限変更を可能としている．権限変更システムコールの終了後，18 行目にて 0x8 を書込み，書込み制限（WD1）を有効化している．

5.2.3　特権奪取攻撃を行う独自システムコールの捕捉

攻撃を行うユーザプロセスによる独自のシステムコール捕捉時の動作結果を**図 7** に示す．図 7 の 3 行目にて，プロセス ID 1661 の独自システムコール（システムコール番号 350）の呼出しを捕捉している．5 行目，および 6 行目にて，PKRS は　0x8 であり，Protection key 1 の書込み制限（WD1）は有効化されている．10 行目において，Protection key 1 の権限情報を格納するページへの書込みが行われた際，ページフォルト（エラー番号 35）を捕捉している．捕捉されたページフォルトは Protection key により保護されたページへの書込み保護違反を示している．

評価においては，特権奪取攻撃の動作確認のため，14 行目から 15 行目にかけて，ページフォルト捕捉後，PKRS に 0x0 を書込み，Protection key 1 への書込み制限（WD1）を無効化し，特権奪取が成功することを確認している．

```
1.  [*] set PKS bit to CR4 at the kernel boot
2.  [  15.723957] CR4: 24446704 PKS value: (1)
3.  24446704 (0b0000 0001 0111 0101 0000 0110 1111 0000)
      1.  bit 24 == 1: protection keys for supervisor mode

4.  [*] set protection key bit to PTE at the user process creation
5.  [  354.860037] PTE flags: 0x8800000006598063 pkey 1
      1.  bit 63–32: 1000 1000 0000 0000 0000 0000 0000 0000
      2.  bit 31–0: 0000 0110 0101 1001 1000 0000 0110 0011
      3.  bit 62-59 == 0001: protection key 1

6.  [*] start setuid (sysnum: 106) system call invocation
7.  [  363.702116] setuid system call invocation
8.  [  363.702179] sysnum: 0x6a  (106)
9.  [  363.702204] PID: user process 1661
10. [  363.702218] PKS PRIV: disable pks: pkey 1
11. [  363.702232] PKS PRIV: access priv information here
12. [  363.702276] read_pkrs for CPU 0: 0x8
13. PKRS: 0x8 (0b1000)
      1.  bit 2 (RD1) == 0: read disable is off for pkey 1
      2.  bit 3 (WD1) == 1: write disable is on for pkey 1

14. [  363.702325] write_pkrs for CPU 0: 0x0
15. PKRS: 0x0 (0b0): PKS has no restriction for pkeys

16. [  363.702383] read_pkrs for CPU 0: 0x0
17. [  363.713991] PKS PRIV: enable pks: pkey 1
18. [  363.714132] write_pkrs for CPU 0: 0x8
19. [  363.714192] read_pkrs for CPU 0: 0x8
```

図 6　PKS 設定，ならびに権限変更許可システムコール捕捉時の動作結果

```
1.  [*] start vulnerable system call (sysnum: 350) invocation
2.  [  364.203190] vulnerable system call invocation
3.  [  364.203227] sysnum: 0x15e (350)
4.  [  364.203275] PID: user process 1661
5.  [  364.203309] PKS PRIV: enable pks currently
6.  [  364.203405] read_pkrs for CPU 0: 0x8
7.  [  364.203496] sys_kvuln01 current privileges 1: uid=33 euid=33 gid=33

8.  [*] process uid, euid, gid, and egid are changed to 0
9.  [*] kernel catches the page fault regarding protection key
10. [  364.204186] PKS: protection keys hw error code 35, pkey 1
11. HW error code 35 (0b100011)
12. Page fault error code bits: from Linux v5.3.18 : arch/x86/include/asm/trap_pf.h
      1.  bit 0 == 1: protection fault, X86_PF_PROT
      2.  bit 1 == 1: write access, X86_PF_WRITE
      3.  bit 5 == 1: protection keys block access, X86_PF_PK
13. [  364.204232] read_pkrs for CPU 0: 0x8
14. [  364.204282] write_pkrs for CPU 0: 0x0
15. [  364.228813] sys_kvuln01 current privileges 2: uid=0 euid=0 gid=0
```

図 7　特権奪取攻撃に利用可能なシステムコール捕捉時の動作結果

6.　考察

6.1　評価に対する考察

提案するセキュリティ機構を適用したカーネルにおいて，特権奪取攻撃に利用可能なカーネル脆弱性を用いた PoC コードに対し，権限情報へ PKS を適切に設定することで，権限情報の改ざんを捕捉し，保護可能なことを示した．評価結果より，提案するセキュリティ機構を用いることで，動作中カーネルにおいて，予め指定した仮想アドレスに対し，権限情報 Protection key の設定，ならびにレジスタ操作により，読書き制限を動的に制御可能である．

評価において，権限情報を格納したカーネルコードの書込み制限の操作を権限変更許可システムコールにのみ許可し，通常システムコールでは権限情報の読込みのみ許可した場合においても，カーネルおよびユーザプロセスの動作には影響を及ぼしていないことを確認した．

6.2　提案手法の考察

提案するセキュリティ機構では，特権奪取攻撃を想定し，

権限情報を PKS による保護対象とした．セキュリティ機構に関するカーネルデータも保護可能であると考えており，アクセス制御ポリシや計算機の資源制御リスト等の攻撃対象となるカーネルデータを調査し，保護可能か検討を進めたい．また，提案するセキュリティ機構では，システムコール単位で権限情報 Protection key を管理し，システムコール発行時に制限操作レジスタの読書きを必要とする．保護対象カーネルデータ毎に適切な制限操作レジスタの操作タイミングは異なると考えており，保護対象を増やす場合，性能負荷への影響を考慮して検討したい．

従来研究として，カーネルのページテーブル分割によるカーネルデータ保護が提案されており [15]，CPU での仮想アドレス変換キャッシュを格納する translation lookaside buffer （TLB） 利用の高速化が必要とされる．PKS は特定レジスタ操作のみで指定した仮想アドレスの読書き制限を制御可能であり，ページテーブル分割，ならびに TLB 管理を伴わず，実装複雑性の削減と性能負荷低減が期待される [5]．提案するセキュリティ機構では，PKS を用いてカーネルデータ保護を実現したが，仮想環境では実性能負荷の評価は困難である．PKS に対応した CPU にてベンチマークソフトウェアを利用した測定を進める予定である．

6.3 限界

提案するセキュリティ機構の実現方式で利用する PKS は， 16 個の Protection key を指定可能である．0 番目の Protection key は PTE の初期値として利用されるため，読書き制限の設定は難しい．保護対象カーネルデータは 15 個の Protection key にて制限管理を行わなければならない．

PKS にて，PTE に指定する Protection key，ならびに制限管理レジスタはカーネルモードにおいて操作可能である．攻撃者がカーネルモードにて任意のコードを実行した場合，提案するセキュリティ機構保護を回避される可能性があり，他のセキュリティ機構との組合せは必要である．CFI は，正規のコード呼出し順に沿う実行か検証し，不正コード呼出し防止に有効である．カーネルへの CFI 適用は，CPU 等ハードウェアセキュリティ機能の無効化防止として，提案するセキュリティ機構と連携可能と考えている．

6.4 移植可能性

提案するセキュリティ機構の他 OS への移植可能性として，Protection key の利用に，OS にてページテーブルによる仮想記憶空間，および PTE の実装が求められるため，FreeBSD 等，Intel CPU に対応した OS では，提案するセキュリティ機構を適用可能であると考えている [16]．

7. 関連研究

MPK を利用したデータの保護

MPK を利用したデータ保護として，ERIM は PSU を用いてアプリケーション向けに保護対象データを異なるユーザプロセスへ分離する手法を提案している [17]．libmpk はユーザプロセスでの保護対象データの操作を柔軟化のため，PSU 管理の抽象化を提案している [18]．また，xMP はカーネルの仮想記憶空間を構成する複数のページをドメイン単位として Protection key 毎に分割し，仮想マシンモニタ（VMM）から管理する機構を提案している [19]．libhermitMPK は複数の Protection key にカーネルコードとデータを分割し保護する機構を提案している [20]．

仮想記憶空間の分離によるカーネルデータの保護

Kernel page table isolation （KPTI）はユーザプロセスによるサイドチャネル攻撃からカーネルデータの参照を防止するため，ユーザモード用の仮想記憶空間に一部のカーネルコードおよびカーネルデータ，大部分をカーネルモード用の仮想記憶空間に配置した [21]．また，Proclocal はユーザプロセス毎にカーネルの仮想記憶空間に占有の記憶領域を割当て [22] ，Address space isolation （ASI）[15] は仮想マシン実行専用のカーネルの仮想記憶空間を備え，仮想化に関するカーネルデータの保護を実現している．

不正コード実行防止

脆弱性を利用した攻撃による不正コード実行に対し，コード呼出し順を検査する control flow integrity（CFI）が提案されている [1]．KCoFI では，カーネルへの CFI 適用のため，独自アーキテクチャとして非同期処理発生時におけるコード呼出順の整合性保存機構を提案している [2]．

7.1 先行研究との比較

先行研究 [15], [19], [20], [22] との比較を**表 3** に示す．Proclocal では，カーネルの仮想記憶空間にユーザプロセス毎の記憶領域を割当て，特定ユーザプロセスのみ参照可能とし，カーネルデータを保護する．また，ASI では専用の仮想記憶空間を生成し，保護対象のカーネル機能に関するカーネルコード，カーネルデータを配置する．Proclocal および ASI の保護対象カーネルデータは，ユーザプロセス毎に管理され，他ユーザプロセスと共有しない．また，カーネルの仮想記憶空間のページ参照制御や切替えが要求され，複雑な実装が必要となる．提案するセキュリティ機構では，PSK を利用し，PTE とレジスタ操作のみでカーネルデータ保護を実現する．複雑なページ操作やカーネルの仮想記憶空間の切替えを伴わず，TLB ミスを抑制し，性能負荷は少ないと考えられる．

libhermitMPK は，Protection key を利用し，2 つの領域（Safe / Unsafe）にカーネルを分割する．実行中カーネルコードは同一領域に属したカーネルデータのみ参照可能とし，カーネルデータを保護する．また，xMP は VMM からゲスト OS カーネルの仮想記憶空間を複数ドメイン領域に分割，ドメイン毎に Protection key を割当て，カーネルコード，カーネルデータ保護を行う．libhermitMPK，お

表 3 カーネルデータ保護手法の比較

機能	Proclocal [22]	ASI [15]	libhermitMPK [20]	xMP [19]	提案方式
保護対象	カーネルデータ	カーネルの特定機能	カーネル全体	カーネル全体	カーネルデータ
制御単位	ユーザプロセス	仮想マシン	カーネル処理	仮想マシン	システムコール
実現方式	ページ参照操作	専用ページテーブル	Unikernel	VMM からのメモリ管理	カーネル内部管理
限界	占有ページ数・カーネルデータ共有の制限	カーネル機能の部分移植性	保護対象とするカーネル機能の安全性	保護対象領域管理の負荷	保護対象カーネルデータ種別数の制限

よび xMP では，カーネル脆弱性とカーネルデータに同一 Protection key を割当てた場合，カーネルデータは改ざんの可能性がある．さらに，xMP では，VMM 上ゲスト OS は MPK 利用を制限される．提案するセキュリティ機構では，システムコール単位で Protection key の書込み制限を制御する．ゼロデイ攻撃など，新たなカーネル脆弱性を介した攻撃に対しても，書込み制限対象システムコールを経由した場合，保護対象カーネルデータを改ざん防止する．

8. おわりに

本稿では，特定のカーネルデータへの読書き制限を制御し，権限情報を保護を可能とするセキュリティ機構を提案した．提案したセキュリティ機構においては，CPU の提供するメモリ保護機構として，PKS を用いた PTE への読書き制限を動作中のカーネルにおいて制御可能とした．実現したセキュリティ機構においては，システムコール単位で権限情報の読書き制限を制御し，攻撃を行うユーザプロセスに対して，権限情報への書込み制限を実現した．

評価においては，提案手法を実現した Linux に対し，特権奪取攻撃に利用可能なカーネル脆弱性を評価用に導入し，ユーザプロセスからの特権奪取攻撃による権限情報の書込み制限を実現可能であることを示した．

謝辞 本研究の一部は，JSPS 科研費 JP19H04109 の助成を受けたものです．

参考文献

[1] Abadi, M., et al.: Control-Flow Integrity Principles, Implementations, and Applications. *Proc. 12th ACM Conference on Computer and Communications Security*, pp. 340–353 (2005).

[2] Criswell, J., et al.: KCoFI: Complete Control-Flow Integrity for Commodity Operating System Kernels. *Proc. 2014 IEEE Security and Privacy*, pp. 292–307 (2014).

[3] Shacham, H., et al.: On the effectiveness of address-space randomization. *Proc. 11th ACM Conference on Computer and Communications Security*, pp. 298–307 (2004).

[4] Yamauchi, T., et al.: Additional kernel observer: privilege escalation attack prevention mechanism focusing on system call privilege changes, *International Journal of Information Security*, vol. 20, pp. 461–473 (2021).

[5] Gleixner, T.: PKS: Add Protection Keys Supervisor (PKS) support, available from https://lwn.net/Articles/826091/. (accessed 2021-08-18).

[6] Weiny, I.: [PATCH RFC PKS/PMEM 00/58] PMEM: Introduce stray write protection for PMEM, available from https://lore.kernel.org/lkml/20201009195033.3208459-1-ira.weiny@intel.com/. (accessed 2021-08-18).

[7] Larabel, M.: Intel Explores Write Protecting Page Tables Using Upcoming PKS Feature https://www.phoronix.com/scan.php?page=news_item&px=Intel-PKS-Write-Protect-PT. (accessed 2021-08-18).

[8] Bonzini, P.: [PATCH] target/i86: implement PKS, available from https://lore.kernel.org/qemu-devel/20210127093540.472624-1-pbonzini@redhat.com/. (accessed 2021-08-18).

[9] Intel Corporation.: Intel(R) 64 and IA-32 Architectures Software Developer’s Manual, available from https://www.intel.co.jp/content/www/jp/ja/architecture-and-technology/64-ia-32-architectures-software-developer-system-programming-manual-325384.html, (accessed 2021-08-18).

[10] Chen, H., et al.: Linux kernel vulnerabilities - state-of-the-art defenses and open problems. *Proc. Second Asia-Pacific Workshop on Systems*, pp. 1–5 (2011).

[11] CVE-2016-4997, available from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-4997. (accessed 2019-05-12).

[12] CVE-2016-9793, available from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-9793. (accessed 2019-05-12).

[13] CVE-2017-1000112, available from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-1000112. (accessed 2019-05-12).

[14] CVE-2017-16995, available from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-16995, (accessed 2019-06-10).

[15] Chartre, A.: Kernel address space isolation, available from https://lwn.net/Articles/813393/, (accessed 2020-06-12).

[16] The FreeBSD documentation project.: FreeBSD architecture handbook, available from https://www.freebsd.org/doc/en_US.ISO8859-1/books/arch-handbook/, (accessed 2019-08-08).

[17] Vahldiek-Oberwagner, A., et al.: ERIM: Secure, Efficient In-process Isolation with Protection Keys (MPK), In: Proceedings of the 28th USENIX Conference on Security Symposium, pp. 1221–1238, (2019).

[18] Park, S., et al.: libmpk: Software Abstraction for Intel Memory Protection Keys (Intel MPK), *Proc. 2019 USENIX Annual Technical Conference*. pp. 241–254 (2019).

[19] Proskurin, S., et al.: xMP: Selective Memory Protection for Kernel and User Space, *Proc. 2020 IEEE Symposium on Security and Privacy*, pp. 563–577 (2020).

[20] Sung, M., et al.: Intra-Unikernel Isolation with Intel Memory Protection Keys, *Proc. 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pp. 143–156 (2020).

[21] Gruss, D. et al.: KASLR is Dead: Long Live KASLR, *Proc. 2017 International Symposium on Engineering Secure Software and Systems*, vol. 10379, no. 3, pp. 161–176 (2017).

[22] Hillenbrand, M.: Process-local memory allocations for hiding KVM secrets, available from https://lwn.net/Articles/791069/, (accessed 2019-08-08).